VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2017 Michal Břemek

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe
Individual Professional Practice in the Company

2017 Michal Břemek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Michal Břemek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: AstrumQ Interactive, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. RNDr. Petr Šaloun, Ph.D.**

Konzultant bakalářské práce: Ing. Daniel Robenek

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Hlučíně, dne 21. 4. 2017

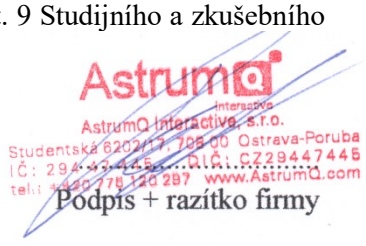
.....

Podpis

Prohlášení zástupce firmy

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě, dne 21. 4. 2017



Podpis + razítko firmy

Abstrakt

Tato bakalářská práce popisuje průběh individuální odborné praxe ve firmě AstrumQ Interactive s.r.o. U této firmy jsem měl možnost vyvíjet mobilní aplikace pro OS Android. Obsahem této práce je popis zadaných úkolů a postup jejich řešení. Následuje ukázka programování a popis samotné aplikace. Dále zde shrnuji své znalosti a dovednosti, které jsem získal během studia a uplatnil je při práci, ale také znalosti a dovednosti, které mi scházeli a které jsem nastudoval až v rámci praxe. Závěr práce je věnován zhodnocení dosažených výsledků a celkového přínosu praxe.

Klíčová slova: Android, mobilní aplikace, Java, programování, OOP, odborná praxe, AstrumQ

Abstract

This bachelor thesis describes the course of individual professional practice in the company AstrumQ Interactive s.r.o. In this company, I was able to develop mobile apps for the Android OS. The content of this work is a description of assigned tasks and the procedure of their solution. Following is a sample of programming and description of the application itself. I also summarize there my knowledge and skills that I gained during my studies and applied them at work, but also the knowledge and skills that I missed and which I had to study within the practice. The conclusion of this work is devoted to the evaluation of the achieved results and to the overall contribution of the practice.

Keywords: Android, mobile applications, Java, programming, OOP, professional practice, AstrumQ

Obsah

Seznam použitých symbolů a zkratek

API	-	Application programming interface [*rozhraní pro programování aplikací*]
DPI	-	Dots per inch [*počet obrazových bodů do jednoho palce*]
JSON	-	JavaScript object notation
MVC	-	Model-View-Controller
MVP	-	Model-Veiw-Presenter
OOP	-	Objektově orientované programování
SDK	-	Software development kit [*soubor nástrojů pro vývoj software*]
XML	-	Extensible markup language [*rozšiřitelný značkovací jazyk*]

Seznam ilustrací

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

Tuto bakalářskou práci jsem se rozhodl vykonat formou absolvování odborné praxe. Tento způsob jsem si zvolil zejména proto, abych získal nové a praktické zkušenosti v oboru informatiky. Chtěl jsem se podílet na skutečných projektech pro skutečné zákazníky. Vyzkoušet si práci v týmu. A také získat nové znalosti a dovednosti, při plnění praktických úkolů. Všechny tyto nově nabité zkušenosti, bych poté chtě využít při budoucím uplatnění.

Tato bakalářská práce se věnuje problematice vytváření mobilní aplikace pro operační systém android. Již od mého nástupu do firmy, ale bylo jasné, že moje programování ještě není na takové úrovni, abych byl sám schopen napsat celou aplikaci. Značná část praxe byla proto věnována studiu potřebných technologií, frameworků a struktur kódu.

2 O společnosti

AstrumQ Interactive, s.r.o. (*https://astrumq.com/*) – společnost zabývající se především: vytvářením webových stránek, vývojem mobilních aplikací a grafickým designem. V současné době se firma může pochlubit aplikacemi, jako jsou: BitOasis, Domaluj si říkanku, MHD jízdenky, Sportuj v Ostravě, MobilKredit a mnoho dalších.

(1) „Společnost AstrumQ Interactive, s.r.o. vyvíjí webové stránky s příběhy a užitečné mobilní aplikace na zakázku. Sídlíme v Ostravě od roku 2012. Naším posláním je zjednodušovat přístup k informacím.“

„Spolupracujeme s klienty malých a středních firem z mnoha odvětví. Zaměřujeme se na vývoj mobilních a webových aplikací zejména v oblastech obchodu, vzdělávání, gastronomie, zábavy a marketingu. Pečlivou analýzou potřeb zjišťujeme možnosti využití webového či mobilního řešení. V rámci spolupráce nabízíme servis mobilních a webových aplikací a odborné poradenství.“

2.1 Pracovní náplň

Oficiální název mojí pracovní pozice je: **Vývojář mobilních aplikací, junior pro Android**. Samotné vykonávání praxe by se pak dalo rozdělit do dvou částí.

V první části, jsem se musel nejprve zaškolit v používání veškerých systémů, používaných ve firmě. Bylo potřeba nastudovat a umět používat potřebné rozšiřující knihovny, frameworky a programovací techniky. Ale hlavně, pořádně se seznámit se samotným vytváření aplikací pro android, protože jsem byl v tomto ohledu stále ještě začátečník.

V druhé části už jsem se přímo podílel na reálném projektu a vyvíjel mobilní aplikaci.

2.2 Sportnect

Dlouhodobý projekt, na kterém se v současné době pracuje. Jedná se o sportovní sociální síť, která má za cíl podporovat a rozvíjet sportovní aktivity.

Aplikace je určena pro širokou sportující veřejnost, která potřebuje efektivní nástroj, jak jednoduše a rychle zorganizovat sportovní událost s přáteli, ve sportovních zařízeních a centrech nebo třeba na dětských hřištích. Aplikace nabízí možnost přidat si své sportovní přátele, komentování sportovních událostí a hodnocením sportovišť i akcí.

3 Analýza projektu

3.1 Webová aplikace

Momentálně je internetu dostupná beta verze webové aplikace a to na stránkách *https://sportnect.com/*. Pro vývojové účely potom na *https://sportnectdev.com/*.

3.2 Mobilní aplikace

Souběžně s webovou aplikací, se pracuje i na nativních aplikacích pro mobilní platformy. V mém případě se tedy jedná o platformu Android. Cílem je vytvořit aplikaci, která bude korespondovat s webovou verzí a později bude i dostupná na Google Play.

3.2.1 Volba úrovně android API

Target SDK by mělo být nastavené na nejnovější API (v současnosti 24, 25). Minimální verze SDK by potom měla být alespoň 14 (Android 4). Moderní aplikace se už nedělají se zpětnou kompatibilitou na nižší verze, protože mnoho nových vlastností přinesl právě Android 4. V kódu by jinak mohlo vzniknout mnoho varování (*warnings*) a bylo by nutné je ošetřit. Navíc, podle statistik zastoupení jednotlivých verzí OS Android, které zveřejňuje Google, už více než 99% aktivních zařízení běží na API 14 a více.

Verze	Kódové označení	API	Podíl na trhu
2.3.3 - 2.3.7	Gingerbread	10	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.9%
4.1.x	Jelly Bean	16	3.5%
4.2.x		17	5.1%
4.3		18	1.5%
4.4	KitKat	19	20.0%
5.0	Lollipop	21	9.0%
5.1		22	23.0%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	4.5%
7.1		25	0.4%

Tabulka 1: Zastoupení verzí android

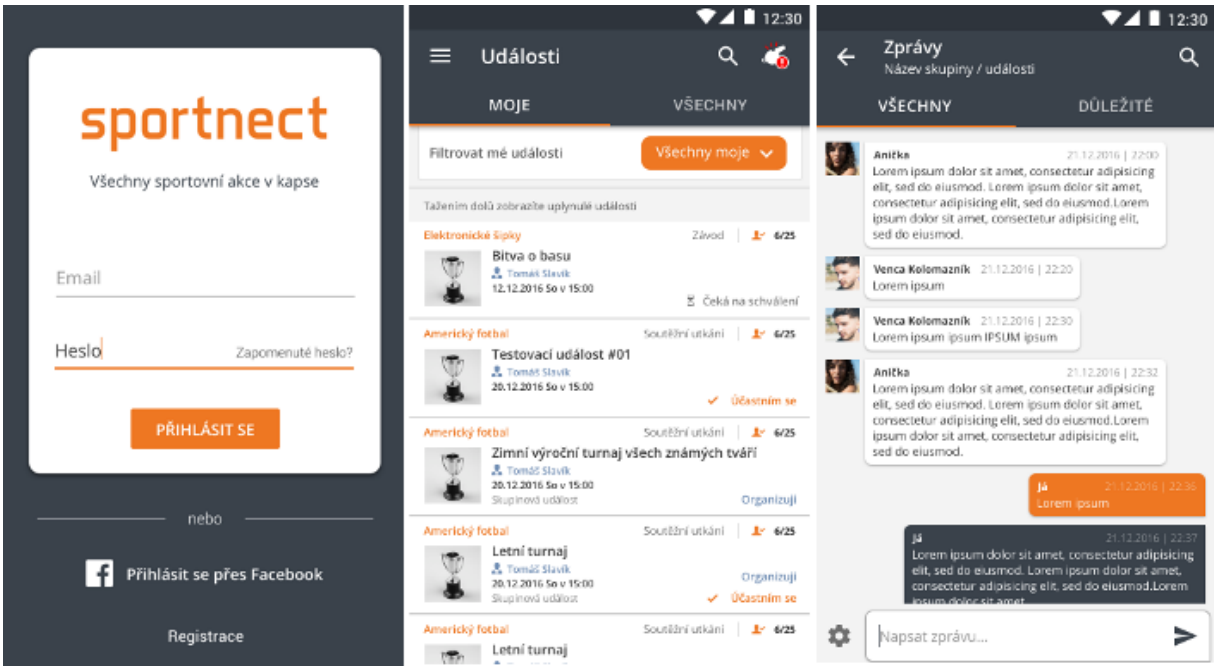
Dostupné z: https://developer.android.com/about/dashboards/index.html

3.2.2 Design

Návrh vzhledu aplikace je nahrán na webové aplikaci InVision (viz 6.4). Odsud budu čerpat při samotné implementaci. K dispozici je zde ucelený náhled na aplikaci. Ke stažení jsou zde veškeré grafické prvky, od palety barev až po obrázky pro různé rozlišení displeje (mdpi, hdpi, xhdpi, xxhdpi a xxxhdpi).

Rozložení jednotlivých prvků a jejich vzhled se bude nastavovat v XML layoutech (viz 5.2.3). Všechny barvy, textové řetězce a obrázky budou uloženy v odpovídajících složkách.

Ukázka grafického návrhu:



Obrázek 1: Grafický návrh aplikace

3.3 Přístupové API

Pro potřeby vývoje používáme webový nástroj Apiary dostupný z *https://apiary.io/*. Definuje způsob komunikace mezi mobilní aplikací a serverem. Je potřeba nad API postavit rozhraní, které s ní bude umět komunikovat. Je zapotřebí správně navázat spojení pomocí http metod get a post (viz Volley 6.2). Data budou posílána ve formátu JSON, je nutné je správně serializovat a deserializovat, k tomu nám může posloužit například knihovna Gson (viz 6.6).

3.4 Prvky androidu

3.4.1 Activity

V aplikaci, každá aktivita obvykle odpovídá jedné obrazovce. Slouží tedy jako hlavní prostředek pro interakci s uživatelem. Aplikace většinou obsahují více aktivit. Vždy, když začne nová aktivita, ta předchozí se zastaví a uloží do zásobníku. Je možné se k ní potom zase vrátit, například pomocí tlačítka zpět. Pro základní komunikaci mezi komponentami slouží Intenty (objekty třídy Intent). Slouží

pro spuštění aktivity nebo služby a dokáže přenášet jednoduchá data. Rozdělujeme je na explicitní a implicitní.

Každá aktivita prochází vlastním životním cyklem a nachází se v jednom ze čtyř stavů. Aktivní, pozastavená, zastavená nebo mrtvá. Systém při přecházení mezi jednotlivými stavy volá callback metody.

3.4.2 Fragment

Fragmenty reprezentují část uživatelského rozhraní v aktivitě. Můžete kombinovat několik fragmentů v jediné aktivitě, nebo je využít ve více aktivitách. Fragmenty mají svůj vlastní životní cyklus a můžete je přidávat nebo odebírat zatímco je jejich aktivita spuštěná.

3.4.3 Adapter

Slouží jako propojení mezi určitým View a daty pro toto View. Poskytuje přístup k datům a také je zodpovědný za vytvoření View pro každou položku z množiny dat. Pro dosáhnutí vyšší paměťové efektivity, má adapter uložená jen navenek viditelná View a ty v těsném okolí. Ostatních se zbavuje a podle potřeby je zase znovu vytváří.

V naší aplikaci budeme potřebovat **ListAdapter**, který slouží k výpisu seznamu prků, mezi kterými můžeme rolovat. A také **PagerAdapter**, který posunuje celé fragmenty, ale vždy si pamatuje jenom tři, tedy ten právě zobrazený a dva okolo.

4 Vypracování

4.1 Activity

4.1.1 LauncherActivity

Jednoduchá aktivita, která se zobrazí ihned po spuštění aplikace. Na obrazovce zobrazí logo aplikace a po určité době spustí LoginActivity. Sama se pak ukončí.

Aktivita obsahuje vnořenou třídu, která dědí ze třídy AsyncTask. Tato třída je zodpovědná za pozastavení aplikace. Jakmile vyprší nastavený čas, tak se zavolá metoda, která spustí další aktivitu.

```
public class Delay extends AsyncTask<Long, Void, Void>{
    @Override
    protected Void doInBackground(Long... params) {
        if(params.length != 0){
            sleep(params[0]);
        }
        onTimeExpired();
        return null;
    }
    private void sleep(Long time){
        try {
            Thread.sleep(time);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Zdrojový kód 1: Asynchronní třída pro odměření stanoveného času

4.1.2 LoginActivity

Slouží pro přihlášení a registraci do systému Sportnect. Je implementovaná podle architektonického vzoru MVP a skládá se tudíž ze tří oddělených tříd.

LoginActivity – zastává roli View. Má na starosti všechny aktivní prvky (EditText, ProgressBar, Button,…). Zpracovává uživatelské vstupy a řídí vzhled okna. Získaná data od uživatele dále posílá ke zpracování Presenteru.

LoginPresenter – kontroluje správnost zadaného hesla a emailu. Poté zašle požadavek na přihlášení serveru a vyčká na odpověď. Přijatá data dále zpracuje a předá zpátky k View.

LoginModel – obsahuje uložené emaily uživatelů, kteří se ke Sportnectu přihlásili z daného zařízení.

4.1.3 MainActivity

Aktivita obsahující boční vysouvací menu (*navigation drawer*). Její hlavní okno obsahuje panel nástrojů (*toolbar*) a volné místo pro zobrazování jednotlivých fragmentů.

Při přepnutí položky v menu se spustí metoda onNavigationItemSelected. Podle toho, která položka byla vybrána, se spustí odpovídající fragment. Pomocí FragmentManageru se fragment zobrazí na požadovaném místě.

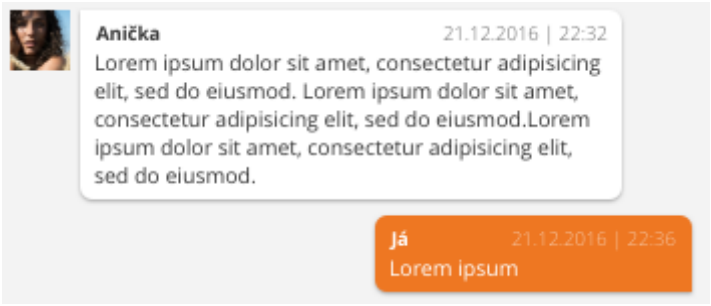
```
private FragmentManager fragmentManager;
private Fragment currentFragment;

@Override
public boolean onNavigationItemSelected(MenuItem item) {
    switch (item.getItemId()){
        case R.id.first_item:
            firstItemSelected();
            break;
        case R.id.second_item:
            secondItemSelected();
            break;
    }
    return true;
}

private void firstItemSelected() {
    currentFragment = new EventMainFragment();
    showCurrentFragment();
}

private void showCurrentFragment() {
    fragmentManager.beginTransaction().replace(
            R.id.content_place, currentFragment, Const.LAST_FRAGMENT
    ).commit();
    drawer.closeDrawer(GravityCompat.START);
}
```

Zdrojový kód 2: Ukázka přepínání stránek v menu

4.1.4 CommentActivity

Umožňuje komunikaci mezi uživateli formou komentářů v bublinách. V aplikaci se aktivita váže ke konkrétním událostem a skupinám.

Komentáře se zobrazují v ListView za pomocí ListAdapteru. Vzhledem k tomu, že se příchozí komentáře v určitých ohledech graficky liší od těch našich. Bylo do xml dokumentu, jednotlivých zobrazovaných prvků, třeba přidat další komponenty. Změnou některých jejich vlastností, nebo třeba úplným skrytím, je poté možné každý komentář upravit podle požadavků. Pro efekt bubliny používám xml kód pro shape.



Obrázek 2: Ukázka zpráv v bublinách

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <corners
        android:bottomLeftRadius="8dp"
        android:topRightRadius="8dp"
        android:topLeftRadius="8dp"/>
</shape>
```

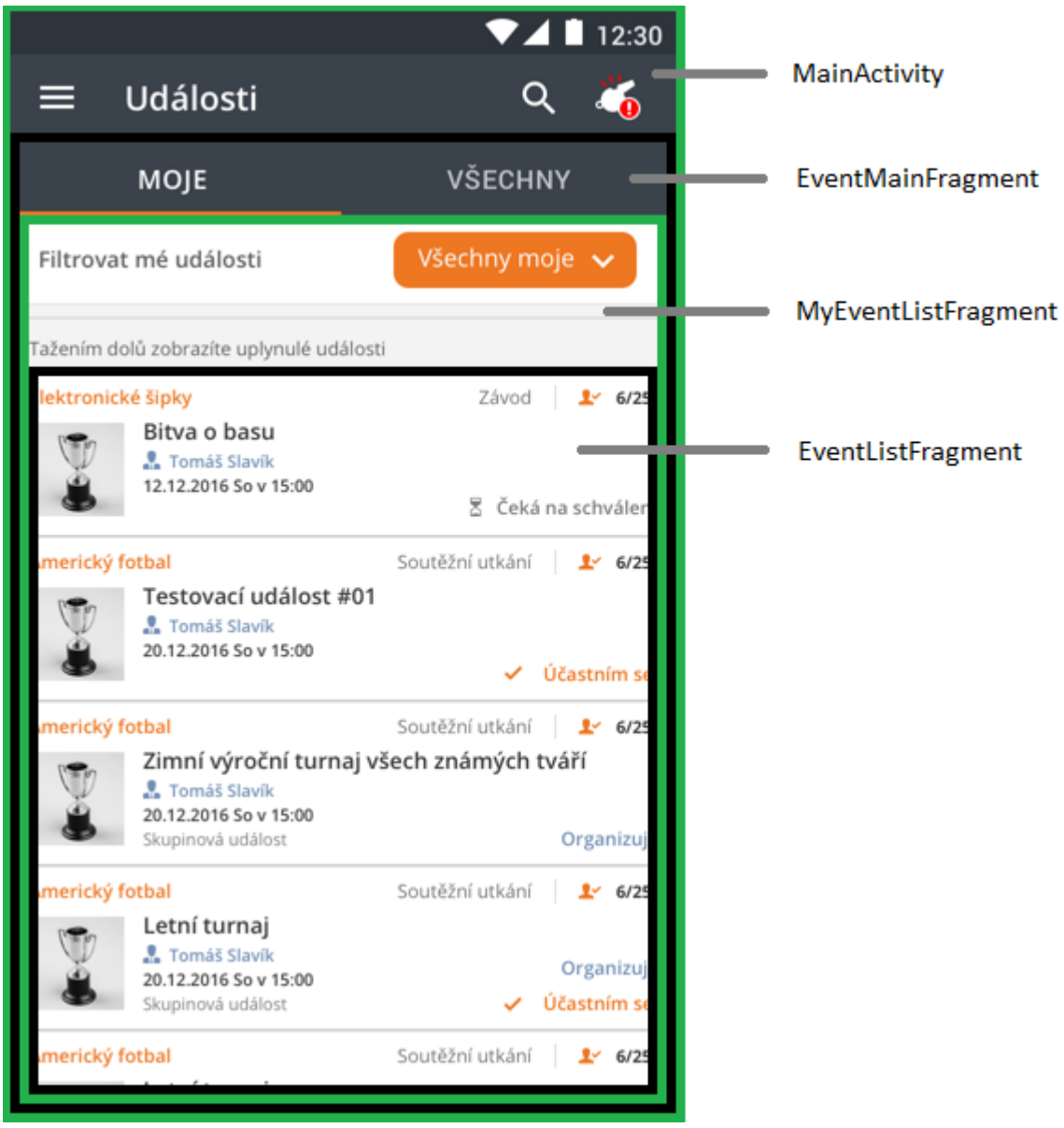
Zdrojový kód 3: Vzor pro ohraničení view

4.2 Fragmenty

4.2.1 EventMainFragment

Zobrazuje výpis událostí. Umožňuje přepínat mezi výpisem „mých“ událostí a výpisem „všech“ událostí.

Fragment je spuštěný v MainActivity. Jeho struktura se skládá ze dvou dalších fragmentů (**MyEventListFragment** a **AllEventListFragment**), mezi nimiž se dá přepínat pomocí PagerAdapteru. Každý z nich ještě obsahuje další fragment (**EventListFragment**) se samotným ListView pro výpis událostí.



Obrázek 3: Struktura okna pro události

4.2.2 GroupMainFragment

Zobrazuje výpis skupin. Strukturou se velmi podobá EventMainFragmentu.

4.3 Adaptéry

V aplikaci používám ListAdaptery v kombinaci se vzorem ViewHolder. Pro různý typ dat vytvářím vždy novou třídu adaptéru.

- EventsListAdapter
- GroupsListAdapter
- MessageListAdapter

Dále používám PagerAdaptery, které drží fragmenty vedle sebe a dovolují mezi nimi přepínat, nebo přejíždět (*swipe*).

- EventsPagerAdapter (přepíná mezi: MyEventListFragment – AllEventListFragment)
- GroupsPagerAdapter (přepíná mezi: MyGroupListFragment – AllGroupListFragment)

4.4 Networking

4.4.1 Client

Stará se o komunikaci se serverem. Vytváří requesty, které následně přidává do fronty třídě Volley. V našem případě vytváříme JsonObjectRequesty. Pro komunikaci používáme http metody GET a POST. Autentizace se provádí pomocí OAUTH2. V hlavičce requestu tedy posíláme parametr obsahující access token, který nám přišel při přihlášení. Na odpovědi vyčkávají dva posluchače responseListener a errorListener. Příchozí zprávy se poté předají zpět třídě, která si o ně požádala.

```
request = new JsonObjectRequest(
    requestMethod, url, params, newResponseListener(), newErrorListener()
) {
    public Map<String, String> getHeaders() {
        Map<String, String> map = new HashMap<>();
        map.put(Param.Auth.ACCESS_TOKEN, User.getAccessToken());
        return map;
    }
};
volley.getRequestQueue().add(request);
```

Zdrojový kód 4: Vytvoření a odeslání nového http požadavku

4.4.2 Volley

Třída vytvořená podle vzoru Jedináček (*Singleton*). To k ní dovoluje přistupovat z celého projektu a může tak řadit veškeré požadavky o webovou komunikaci do fronty. O frontu požadavků se stará stejnojmenná knihovna Volley. Kód celé třídy viz Zdrojový kód 5.

```
public class Volley {
    private static Volley instance;
    private RequestQueue requestQueue;
    private static Context context;

    public synchronized static Volley getInstance(Context context) {
        if (instance == null) {
            instance = new Volley(context);
        }
        return instance;
    }

    private Volley(Context context) {
        this.context = context;
        requestQueue = getRequestQueue();
    }

    public RequestQueue getRequestQueue() {
        if (requestQueue == null) {
            requestQueue =
com.android.volley.toolbox.Volley.newRequestQueue(context);
        }
        return requestQueue;
    }
}
```

Zdrojový kód 5: Třída Volley podle vzoru Singleton

4.5 Ostatní třídy

4.5.1 App

Třída dědí ze třídy Application. Je vytvořená jako jedináček. Pomocí metody getContext pak globálně poskytuje context pro celý projekt.

4.5.2 Konstanty

Veškeré konstanty jsou zde uloženy v jediném balíčku. Rozdělené do několika abstraktních tříd.

ApiUrl – obsahuje url adresu se kterou komunikujeme a konkrétní cesty ke všem stránkám webu.

Params – obsahuje veškeré názvy parametrů, potřebných pro webovou komunikaci.

Const – obsahuje další konstanty používané v aplikaci.

4.5.3 Objects

Balíček tříd, jejichž struktura se shoduje se strukturou příchozích zpráv Jsonu. Pomocí knihovny Gson se pak data správně namapují na patřičné třídy.

4.5.4 ResponseListener

Rozhraní (*Interface*) definující dvě funkce: onResponse a onNetworkError. Všechny třídy, které komunikují pomocí třídy Client musí implementovat toto rozhraní.

4.6 Testování

Testování probíhalo na reálných zařízeních, ale vzhledem k tomu že aplikace ještě není zcela hotová a v dohledné době se ještě nechystá její zveřejnění. Nebyla aplikace prověřena, jak se bude chovat v ostrém provozu. Testování probíhalo ručním odzkoušením na zařízení, vždy když vznikl nový kód.

Samsung Galaxy S4	API 17
Xperia Z3 Compact	API 24

Tabulka 2: Mobilní zařízení použitá při testování

5 Uplatněné znalosti, získané během studia

5.1 Programování

Při studiu jsem se nejprve seznámil se základy strukturovaného programování, se základy jazyka C++ a se základy programových konstrukcí. Naučil jsem se vytvořit a odladit jednoduchý program, využívat datové struktury, jako je například pole, napsat rekurzívní funkci, využívat třídící a vyhledávací algoritmy.

Další úrovní bylo seznámení se s objektově orientovaným programováním a rozvinutí znalostí do oblasti datových struktur. Naučil jsem se analyzovat zadaný problém z pozice OOP, využívat binární stromy a hašovací tabulky a také posoudit efektivitu zvoleného řešení daného problému. Objektově orientované přístupy jsou nezbytnou součástí odborné přípravy každého absolventa informatiky.

Od C++ se dostáváme k tvorbě objektově orientovaných aplikací v rámci platformy Java a v rámci platformy .NET Framework se zaměřením na práci v jazyce C#. Naučil jsem se zde, jak vyvíjet aplikace za použití rozhraní a znovupoužitelných komponent, jak analyzovat, navrhnout a implementovat aplikace nad danými platformami a ještě i práci s kolekcemi, soubory a proudy.

V neposlední řadě jsem se při studiu, co se programování týče, seznámil se základy tvorby aplikací pro mobilní zařízení. Dokážu posoudit vhodnost použité platformy pro zamýšlenou aplikaci, navrhnout a implementovat aplikaci pro mobilní zařízení a rozhodnout jakým způsobem danou aplikaci distribuovat. Časem jsem se dostal i k pokročilejším technikám vývoje aplikací pro mobilní zařízení, zejména pro mobilní telefony na platformě Android. Kde se pro vývoj používalo vývojové prostředí Android studio, podrobněji popsáno v následujícím oddílu 5.2.

5.2 Android studio

Oficiální vývojové prostředí od společnosti Google a JetBrains, vytvořené speciálně pro nativní vývoj aplikací pro platformu Android. Prostředí je postaveno nad softwarem IntelliJ IDEA. Android studio je zcela zdarma a je volně ke stažení pro operační systémy Windows, MacOS a Linux.

Základem každého projektu jsou tyto části: Manifest, zdrojový kód aplikace, Resources a Gradle. Na každou z nich se teď podíváme trochu blíž.

5.2.1 Manifest

Android manifest je opravdu základní součástí každého projektu. Každá Android aplikace musí obsahovat soubor AndroidManifest.xml, který se nachází v kořenovém adresáři projektu. Tento soubor musí obsahovat veškeré základní informace o aplikaci. Bez nich by nebylo možné takovou aplikaci spustit. Tyto informace potom využívá služba Google Play.

Musí zde být uveden název, ikona a téma (*theme*) aplikace. Pro jaké verze Androidu je aplikace určená, tj.: Target SDK version a Min SDK version. Dále zde musí být všechny použité Activity a veškeré potřebné oprávnění (*permissions*).

5.2.2 Java

Je objektově orientovaný programovací jazyk používaný i k programování pro Android, ale v tomhle případě, mám spíše na mysli samotnou strukturu zdrojového kódu jako celek. Tedy veškeré Activity a třídy obsazené v balíčcích (*packages*).

Activity - Hlavní a základní část programu. Je to třída, která dědí z Activity nebo AppCompatActivity. Nejčastěji se tímto pojmem označuje třída, která je napojená na konkrétní View (XML layout), které je viditelné uživateli. Tato Activita je schopna přijímat a zpracovávat UI události a dále pak vykonávat komplexní úlohy.

5.2.3 Resources

Zdroje, nebo také prostředky Android aplikací. Skládají se z obrázků, textových řetězců, různých XML souborů a mnoha jiných souborů. Jsou oddělené od samotného kódu - patří do adresáře *res*. Podle svého typu se dále umísťují do různých podadresářů (drawable, layout, menu, values, raw, …). S prostředky aplikace souvisí i vygenerovaná třída *R.java*, kde jsou seskupeny identifikátory. S její pomocí lze k Resourcům přistupovat z kódu nebo se na ně odkazovat v layoutech.

Každý Resource může ještě obsahovat upřesňující modifikátor (*qualifier*) - to je vlastně jen pomlčkou oddělený speciální řetězec přidaný za jeho název. Tento modifikátor pak určuje, který přesně požadavek bude použit. Např.: lokalizace (*-cs, -en, -fr, ...*), orientace zařízení (*-port, -land*), hustota pixelů (*-mdpi, -hdpi, -xhdpi, ...*) nebo verze androidu (*-v3, -v15, ...*). Lze kombinovat i více koncovek za sebou.

Drawable – obsahuje obrázky, popřípadě jiné XML soubory.

Values – obsahuje soubory, jako jsou: *strings.xml, styles.xml* nebo *colors.xml*

Layout – obsahuje XML soubory definující grafický vzhled a rozložení uživatelského rozhraní. V Android studiu lze navrhovat design aplikace buď v XML, nebo v designovém módu. Mezi těmito módy lze kdykoli přepínat. V designovém i textovém módu umožňuje studio automaticky zobrazit náhled okna.

5.2.4 Gradle

Android Studio je spjaté s buildovacím nástrojem Gradle. Kombinuje flexibilitu Antu a systém závislostí Mavenu. Umí kompilovat, testovat, publikovat a přenášet konfiguraci projektu. A hlavně, díky němu lze i snadno importovat nezbytné knihovny.

5.3 Model-View-Presenter

Mluvíme-li o MVP, pak se bavíme o způsobu jak rozdělit interní logiku od grafického rozhraní. Nejpřesnější termín pro označení MVP je „architektonický vzor“, ale někdy se také můžeme setkat s pojmy, jako jsou prezentační vzory nebo přímo architektura. Pro úplnost, dalším podobným vzorem je i MVC (Model-View-Controller).

Základním principem MVP je tedy rozdělení aplikace, nebo obecně objektu, na základní tři části. Na Model, který se stará o data a základní business logiku. View, která má na starost uživatelské rozhraní. A nakonec Presenter, jehož úkolem je správná funkcionalita a interní logika. Takto rozdělený

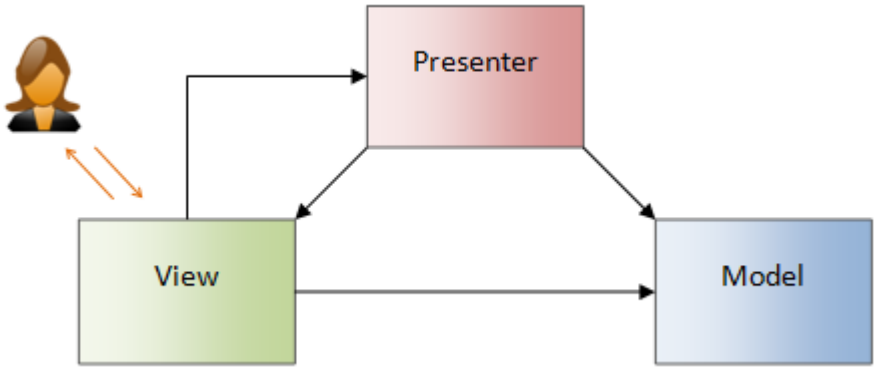
kód na tři části (nebo třídy) bude poté mnohem čistší, bude se dát snadněji odladit a pravděpodobně bude méně náchylný na vznik chyb. Velkou výhodou je také modulárnost. Měli bychom být schopni zaměnit jednotlivé části za jiné, aniž bychom tím ovlivnili chování zbylých dvou.

Model – uchovává potřebná data a vykonává nad nimi základní logiku. (Modely, které obsahují pouze datové struktury bez jakékoli logiky, jsou považovány za antivzory!) Důležité ale je, že u Modelu nesmí existovat přímá vazba na ostatní dvě komponenty. Při implementaci Model pouze poskytuje přístupové metody k potřebným datům.

View – má na starost interakci s uživatelem. Definuje vzhled uživatelského rozhraní a také jednotlivé vstupy a výstupy. View má vazby na Presenter i na Model. Z Modelu si může brát potřebná data k zobrazení. A naopak data získaná z uživatelského vstupu předává Presenteru pro další zpracování.

Presenter – stará se o hlavní logiku a funkcionalitu aplikace. V praxi tedy zpracovává data, která mu byla zaslána z View. Na základě jejich zpracování potom vysílá požadavky buď do Modelu nebo zpět do View. Presenter tudíž obsahuje vazby na obě tyto komponenty.

Čerpáno z článku na webu Zdrojak.cz (2)



Obrázek 4: Schéma architektonického vzoru MVP

6 Dovednosti a znalosti, získané na praxi

6.1 Git

(3) Git je verzovací systém. Sleduje změny v souborech a pomáhá nám vytvářet jejich jednotlivé verze. Primárně je používán při vývoji softwaru, ale jde využít i pro sledování jakýchkoliv jiných souborů. Při práci ve více lidech nám usnadňuje vzájemnou koordinaci. Každému vývojáři dává jednoduchý přístup k celé historii projektu (každý má lokální kopii)**.** Můžeme se libovolně v historii přesouvat a máme přehled o tom, kdo vytvořil jakou část kódu a co se konkrétně změnilo. Díky větvení můžeme vytvářet novou funkcionalitu, aniž by to mělo jakýkoliv dopad na zbytek projektu.

Git je mocná podpora pro nelineární vývoj. Podporuje rychlé vytváření větví (*branch*) a rychlé slučování (*merge*). Toto sloučení nemusí vždy proběhnout automaticky, někdy to zkrátka není možné, a musí být provedena ruční editace. Nové verze jsou vytvářeny potvrzením nově vzniklých změn (*commit*). Tyto změny jsou zatím uloženy pouze lokálně, jakmile je pak uložíme na server (*push*) už je nebudeme moct nijak upravovat. Naopak můžeme verze ze serveru i stahovat (*fetch/pull*) například když je tam nahraje někdo další.

SourceTree (4) – multiplatformní aplikace pro práci se systémy Git a Mercurial. Poskytuje grafické UI pro jednoduchou práci s projektem a jeho uložištěm. Obsahuje nástroje pro vizualizaci a navigaci v nelineární historii vývoje projektu. Také nabízí snadné napojení na uživatelské účty služeb GitHub, BitBucket nebo Kiln.

6.2 Volley

(5) Knihovna Volley je vyvíjená společností Google. Slouží pro rychlou a snadnou komunikaci přes internet v systému Android. Mezi její výhody patří automatické plánování síťových požadavků, umožnění více souběžných připojení k síti, poskytnutí paměťového cachování a také umožňuje zrušit pouze jednu nebo všechny žádosti. Samotný Google jí využívá u svých služeb jako například YouTube, Google play a jiné.

I samotné použití je celkem jednoduché. Není třeba vytvářet nová vlákna pomocí AsyncTask, jednotlivé požadavky se totiž vyřizují asynchronně na pozadí. Je třeba ale vytvořit statickou třídu VolleySingleton (návrhový vzor jedináček), na kterou budeme moct globálně posílat naše požadavky (*requests*) přidáváním do fronty.

6.3 Active Collab

Online nástroj pro podporu řízení projektů. Nabízí funkce jako je docházkový systém, řízení a přehled úkolů, spolupráce, sledování času a fakturace.

Osobně jsem Active Collab využíval právě pro zápis svojí docházky a pro přehled všech mých otevřených či dokončených úkolů. Tyto informace jsou zejména důležité pro vedoucího projektu.

6.4 InVision

Online prohlížečový nástroj, pro tvorbu interaktivních grafických návrhů aplikací a webových stránek.

Jako vývojář se zde můžu podívat na návrh celé aplikace. Jak budou stránky v budoucnu vypadat a jak budou fungovat. Snadno si zde můžu dohledat všechny designové prky jako jsou obrázky, barvy, písmo nebo rozložení stránky a stáhnout si je pro skutečný projekt.

6.5 Clean code

Čistý kód je technika programování. Je to hranice mezi dobrými a špatnými programátory. Její základy stanovil a sepsal **Robert C. Martin** právě v knize Clean Code. (6)

Ve firmě kde jsem pracoval, byl kladen velký důraz na kvalitní a hlavně také „čistý“ kód. Dostal jsem tedy za úkol přečíst si již zmíněnou knihu Clean Code a řídit se její filozofií. Samotná kniha má něco přes 400 stran a proto je pro mě velice obtížné jí zde ve stručnosti shrnout. Pro začátek zde můžu citovat několik definic, uvedených v knize, o tom co to vlastně je čistý kód.

„Čistý kód je jednoduchý a přímočarý. Čistý kód se čte jako dobře napsaná próza. Čistý kód nikdy nezatemňuje záměr návrháře, ale je plný břitkých abstrakcí a přímých toků řízení.“, **Grady Booch**, autor knihy „Object Oriented Analysis and Design with Applications“.

“Že pracujete s čistým kódem, poznáte podle toho, že každá procedura, kterou procházíte, se ukáže být tím, co jste do značné míry předpokládali. Kód můžete označit za nádherný, když vypadá, jako kdyby byl použitý jazyk pro daný problém stvořen.”, **Ward Cunningham,** tvůrce „Wiki“, spolutvůrce „eXtrémního Programování“, myšlenkový představitel OOP.

A teď už se v bodech pokusím vyzdvihnout hlavní body či myšlenky obsažené v knize.

- Používejte popisná jména
- Názvy funkcí by měly sdělovat, co dělají
- Pokaždé když vás napadne výstižnější název – refaktorujte! (Moderní vývojová prostřední poskytují spousty pomoci s refaktorizací)
- Dělejte funkce co nejmenší (měly by provádět jen jednu věc)
- Dávejte funkcím co nejmenší počet argumentů
- Vytvářejte kód tak aby se dal číst odshora dolů
- Neopakujte se (žádný zdvojený kód)
- Pište kód tak, aby ho nebylo nutné doplňovat komentářem (komentáře nevyváží špatný kód)
- Vyhýbejte se zakomentovanému kódu
- Dejte přednost výjimkám před vracením chybových kódů
- Mějme kód pokrytý testy
- Třídy by měli být malé
- Zanechejte kód čistší, než jaký byl na začátku
- Pokud nebudete svůj kód udržovat, začne postupně degenerovat a bude čím dál tím těžší doplnit ho o novou funkcionalitu

6.6 Gson

(7) Javovská knihovna sloužící k usnadnění převodu objektu na jeho JSON reprezentaci (*serializace*) a opětovný převod zpět na objekt (*deserializace*).

Gson lze využít například při převodu primitivních datových typů. Ale jeho hlavní výhoda se ukáže až u obsáhlejších objektů. Při převodu z JSON-u na objekt dokáže Gson správně namapovat hodnoty do vnitřních proměnných daného objektu, ale jen za předpokladu že se název vnitřní proměnné shoduje s patřičným klíčem v JSON-u. Tímto způsobem dokážeme v jednom kroku namapovat i vícenásobně vnořené objekty.

Příklad použití v kódu:

```
Gson gson = new Gson();
gson.toJson(5);          //->"5"
int a = gson.fromJson("5", int.class);
```

Zdrojový kód 6: Ukázka použití knihovny Gson

6.7 Picasso

(8) Velice jednoduchá knihovna pro manipulaci s obrázky. Umožňuje nám je načítat z resources, ze složky nebo je přímo stahovat z internetu a rovnou je zobrazit v příslušném ImageView. Načtené obrázky můžeme podle potřeb transformovat. Také můžeme nastavit PlaceHolder během stahování z internetu. Knihovna je dobře optimalizovaná. Využívá minimální množství paměti a automaticky se stará o cache paměť.

Příklad použití v kódu:

```
Picasso.with(context).load(url).into(imageView);
```

Zdrojový kód 7: Ukázka použití knihovny Picasso

6.8 Butter knife

(9) Knihovna pro snadné navázání Resources a jednotlivých View z Layoutu, na jejich korespondující třídní proměnné. Pomocí anotací a odpovídajících Id nadeklarujeme jednotlivé proměnné a poté už je stačí jenom navázat pomocí jediného příkazu. Tento způsob nám dovoluje nahradit dosavadní způsoby pomocí metod *findViewById(id)* nebo třeba *getDrawable(id)*. (8)

Příklad použití v kódu:

```
@BindDrawable(R.drawable.img) Drawable img;
@BindView(R.id.text) TextView text;       //deklarace
ButterKnife.bind(this);                   //v kódu
```

Zdrojový kód 8: Ukázka použití knihovny Butter knife

7 Závěr

Během mé odborné praxe ve společnosti AstrumQ Interactive, s.r.o. jsem si vyzkoušel práci v týmu na rozsáhlejších projektech. Byl jsem zaškolen v používání interních systémů a poté jsem se také naučil pracovat s potřebnými technologiemi a knihovnami. Účastnil jsem se vývoje aplikací i jejich testování.

Mým hlavním cílem ve firmě, bylo vytvořit mobilní aplikaci Sportnect pro android. Jedná se o sportovní sociální síť, která má za cíl podporovat a rozvíjet sportovní aktivity. Než jsem nastoupil, už se pracovalo na její webové verzi, která je v současnosti stále ještě ve vývoji. Při vývoji aplikace jsem čerpal z vyhotoveného designového návrhu. Hlouběji jsem se seznámil se strukturou a principy programování v Android studiu. Nicméně vzhledem k rozsahu projektu, nebylo možné aplikaci včas dokončit.

S využitím poznatků získaných během praxe, jsem nyní schopen vytvořit téměř jakoukoli aplikaci pro platformu android. Můžu říct, že jsem se více zdokonalil v programování i v OOP. Jako další přínos pak považuji i samotnou práci v týmu a osobně předpokládám, že mi to může pomoct i při hledání budoucího uplatnění, ať už to bude opět programování pro android, nebo pro cokoliv jiného.

Použitá literatura

1. **AstrumQ.** Vize. *astrumq.com.* [Online] AstrumQ Interactive, s.r.o. [Citace: 2. 3. 2017.] https://astrumq.com/astrumq/.

2. **Bernard, Borek.** MVC a další prezentační vzory. *zdrojak.cz.* [Online] [Citace: 27. 2. 2017.] https://www.zdrojak.cz/serialy/mvc-a-dalsi-prezentacni-vzory/.

3. **Git.** *git.* [Online] [Citace: 7. 3. 2017.] https://git-scm.com/about/branching-and-merging.

4. **Atlassian.** *SourceTree.* [Online] [Citace: 7. 3. 2017.] https://www.sourcetreeapp.com/.

5. **Segato, Gianluca.** An Introduction to Volley. [Online] [Citace: 23. 3. 2017.] https://code.tutsplus.com/tutorials/an-introduction-to-volley--cms-23800.

6. **Martin, Robert C.** *Čistý kód.* [překl.] Jiří Berka. Brno : Computer Press, a. s., 2009. 978-80-251-2285-3.

7. **Google, Inc.** Gson. [Online] [Citace: 22. 3. 2017.] https://github.com/google/gson.

8. **Square, Inc.** *Picasso.* [Online] [Citace: 19. 3. 2017.] http://square.github.io/picasso/.

9. **Wharton, Jake.** *Butter Knife.* [Online] [Citace: 18. 3. 2017.] http://jakewharton.github.io/butterknife/.